2015-11-03.kl.14-19

Examination - Ada

TIPS 1:

Läs igenom ALLA uppgifterna. Välj den du känner är lättast först. Det kan gärna ta 10-20 minuter. Försök skriva saker som kan vara problem i uppgifterna. Är det något du absolut kommer att fastna på så kanske det är fel uppgift att ge sig på. Tiden du lägger på att noga läsa uppgifterna tjänar du in på att välja rätt uppgift.

TIPS 2:

Kolla ibland till kommunikationsfönstret. Det kan ha kommit information till alla utan att ni skickat in en fråga. Kanske gäller det dig också (d.v.s. den uppgift du jobbar med).

TIPS 3:

Sista timmen är vi normalt sett lite tydligare och ger lite "bättre" återkoppling på era fel. Före detta kan man få högre betyg, så då får ni själva leta reda på vad som skall rättas till. Under hela tentan ger vi dock information om vilket symptom felet uppvisar om vi kompletterar. Om ni är nära G tidigt på tentan kan detta kanske ge G i slutet (vi meddelar detta i så fall). Har ni fått ett meddelande om att detta kan ge betyg 3, men inte högre i kombination med andra uppgifter och ni är nöjda med betyg 3 och vill gå kan ni alltid skicka ett meddelande till oss så blockerar vi de övriga uppgifterna så att du kan få ditt betyg och gå.

TIPS 4:

Om ni har problem med kompilator, Emacs eller annat som INTE har med uppgifterna att göra, räck upp handen så kommer en assistent. Detsamma gäller om hur man kopierar givna filer
" cp given_files/* . " eller liknande.

Frågor om själva uppgifterna tar vi i första hand via tentasystemet.
Vi hinner inte svara på frågor de sista 10 minutrarna på tentan. Då ägnar vi all tid åt att rätta uppgifter för att alla skall hinna få svar innan ni går hem. Om det är så att ni skickar in en uppgift precis i slutet av tentan hinner vi förstås inte, men då får ni gärna skicka ett mail för att få reda på hur det gick.

Vi rapporterar in resultaten så fort vi kan efter tentan. Det tar ett par dagar med pappershantering efter detta så räkna med att de är i LADOK om ca en vecka. Kolla med era kompisar om de fått poäng innan ni skickar mail till oss.

Betygsgränser:

1 uppg	19:00	Betyg 3
2 uppg	18:00	Betyg 4
2 uppg	16:30	Betyg 5
3 uppg	18:00	Betyg 5

Bonustid från laborationerna tillkommer till dessa tidsgränser.
Lycka till med tenterandet och hoppas att alla får G på minst en uppgift idag.

M.v.h.
/Torbjörn (examinator)

2015-11-03.kl.14-19

Uppgift 1 (Tyngsta bilen)

Skriv ett program som läser in fem *bilar* och skriver ut den tyngsta bilen på skärmen. Bilarna skrivs in på egna rader (max 60 tecken) och har följande format:

```
REGNUM Totalvikt modell färg
```

Du kan utgå ifrån att alla strängar endast består av tecknen 'A'-'z' och ' ' (blanksteg).

Körexempel:

```
Mata in fem bilar:
PXK013 950 kg Honda Accord Black
NAB341 1630 kg Volvo S80 Brown
TFX232 1950 kg Opel Combo AB11 White
DRV555 1441 kg Ford Focus Green
RTJ112 1091 kg Citroen C3 Blue

Tyngsta bilen var TFX232 som väger 1950 kg.
```

KRAV: Inläsningen av **en** bils relevanta data skall ske i **ett** underprogram.
KRAV: Programmet skall vara generellt gjort, så att det t.ex. är enkelt att ändra från fem till sex bilar.

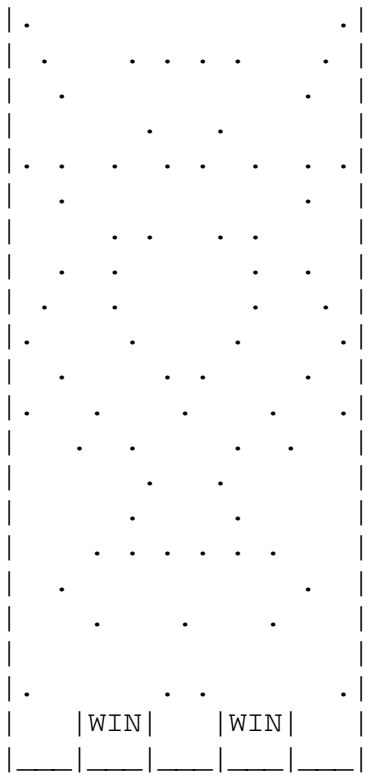
2015-11-03.kl.14-19

Uppgift 2 (Galtonspel)

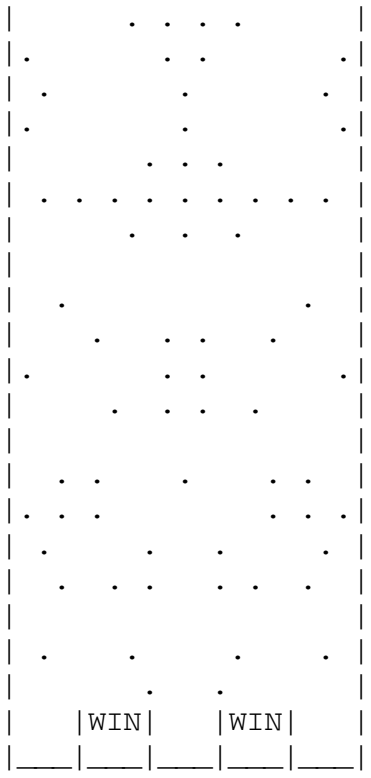
Tänk dig en grund lång trälåda med många små spikar i botten. Om man ställer upp lådan mot en vägg så kan detta vara ett *galtonspel*. I detta spel släpper spelaren en liten kula längst upp. Kulan studsar sedan på spikarna på vägen ner och landar i ett av fem fack. Målet med spelet är att försöka få kulan att landa i de fack som är markerade för vinst.

Skriv ett program som slumpar ut hur galtonspelet ser ut. I terminalen skall spelet ha 20 rader och 19 kolumner (exklusive de två yttre kolumnerna och raderna för vinstfacken längst ner). Det skall vara exakt 70 spikar (ritas ut som punkter). Två spikar får inte sitta precis intill varandra om de sitter på samma rad (då kan kula råka fastna där). Spikmönstret skall vara symmetriskt, det skall alltså vara speglat längs den 10:e kolumnen. De två understa raderna med vinstfacken ser alltid ut på samma sätt oavsett spikmönstret.

Körexempel 1:



Körexempel 2:



2015-11-03.kl.14-19

Uppgift 3 (Dinopaket)

En dinosaurie är en typ av utdött djur som förekom på jorden för ca 230 miljoner år sedan till 66 miljoner år sedan. Du skall i denna uppgift skapa paketet *dino_package* som innehåller en datatyp *Dino_Type* som representerar en dinosaurie och ett par rutiner som använder datatypen.

För att representera en dinosaurie behövs följande data:

- Artnamn (maximalt 100 tecken behövs).
- Vikt (i kg).
- Höjd (i meter).
- Något som talar om om dinosaurien är en kött- eller växtätare.
- Tidsperiod(er).

Med tidsperioder menar vi under vilka tidsperioder som dinosauriearten förekom. Totalt sett förekom alla dinosaurier under tidsperioderna *Trias, Jura* och *Krita*. I princip skulle det kunna vara många fler tidsperioder än tre, tänk på detta då du väljer datatyp för tidsperioden. Det vanliga var alltså att en viss dinosaurieart förekom under en sådan tidsperiod men det fanns arter som existerade under två perioder och några (mycket ovanliga) som existerade under alla tre tidsperioder.

Datatypen skall vara privat i ditt paket. Datatypen får innehålla fler delar än vad som listas ovan.

Det finns ett givet huvudprogram *dino_hp.adb* som du ska testa ditt program med. Programmet utgår ifrån att det finns tre underprogram i paketet:

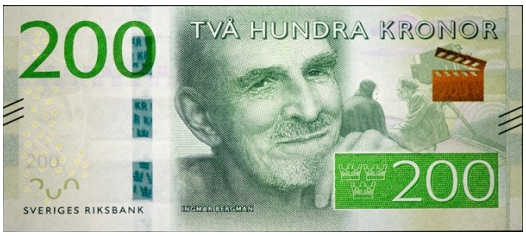
- En procedur Get som kan läsa in till din dinosaurietyp från tangentbordet.
- En procedur Put som kan skriva ut din dinosaurietyp till skärmen.
- En funktion "<" som jämför två dinosaurier och returnerar sant om den vänstra är *mindre än* den högra och falskt annars. Du får själv avgöra exakt hur jämförelsen skall gå till, men den skall använda både dinosauriernas vikt och höjd och du skall med en kommentar vid funktionsdeklarationen i dino_package.ads beskriva exakt hur jämförelsen sker.

Paketet får innehålla fler underprogram än vad som listas ovan.

KRAV: Du får inte ändra på dino_hp.adb.
TIPS: Det finns ett litet program som visar hur man med satsen *get_line* kan läsa in ett namn till datatypen *String*. Programmet finns i mappen given_files och heter *test_get_name.adb*.

2015-11-03.kl.14-19

Uppgift 4 (Svenska mynt och sedlar)



Under oktober månad fick vi ett par nya sedlar i Sverige. Det som är nytt är att vi får en tvåhundrakronorssedel. Om man tittar framöver kommer vi även att få nya mynt och då får vi tillbaka tvåkronan. Om vi tittar på vilka valörer vi kommer att ha i mynt och sedlar så har vi alltså följden:

1, 2, 5, 10, 20, 50, 100, 200, 500, 1000

Det är inte svårt att luska ut vad nästa valör i serien skulle vara (om du inte ser det direkt, kolla på körexempel 1 nedan). Skriv ett program som låter användaren mata in N och skriver ut alla valörer (och "tänkta" valörer) som är mindre än eller lika med N. N är alltid ett positivt heltal. Utskriften av valörer skall ske i ett underprogram som heter Put. Upprepningar i din lösning skall ske med **rekursion**.

Körexempel 1:

```
Mata in N: 2000
Valörerna är: 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000
```

Körexempel 2:

```
Mata in N: 99
Valörerna är: 1, 2, 5, 10, 20, 50
```

Körexempel 3:

```
Mata in N: 110000
Valörerna är: 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000,
5000, 10000, 20000, 50000, 100000
```